Importing required Libraries

In [9]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import binom, norm, ttest_1samp, ttest_ind, ttest_rel, chi2, chisquare, chi2_contingency, stats
from scipy.stats import f_oneway, kruskal, levene, shapiro, pearsonr, spearmanr
from statsmodels.graphics.gofplots import qqplot
import statsmodels.api as sm
```

In [34]:

```
campaign = pd.read_csv('/content/campaign - campaign.csv')
shopping = pd.read_csv('/content/shopping.csv')
```

Analyses on Campaign Dataset

In [35]:

```
campaign.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2239 entries, 0 to 2238
Data columns (total 27 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2239 non-null   int64
 1   Year_Birth           2239 non-null   int64
 2   Education            2239 non-null   object
 3   Marital_Status       2239 non-null   object
 4   Income               2239 non-null   object
 5   Kidhome              2239 non-null   int64
 6   Teenhome             2239 non-null   int64
 7   Dt_Customer          2239 non-null   object
 8   Recency              2239 non-null   int64
 9   MntWines             2239 non-null   int64
 10  MntFruits            2239 non-null   int64
```

```
 10  MntFruits            2239 non-null   int64
 11  MntMeatProducts      2239 non-null   int64
 12  MntFishProducts      2239 non-null   int64
 13  MntSweetProducts     2239 non-null   int64
 14  MntGoldProds         2239 non-null   int64
 15  NumDealsPurchases    2239 non-null   int64
 16  NumWebPurchases      2239 non-null   int64
 17  NumCatalogPurchases  2239 non-null   int64
 18  NumStorePurchases    2239 non-null   int64
 19  NumWebVisitsMonth    2239 non-null   int64
 20  AcceptedCmp3         2239 non-null   int64
 21  AcceptedCmp4         2239 non-null   int64
 22  AcceptedCmp5         2239 non-null   int64
 23  AcceptedCmp1         2239 non-null   int64
 24  AcceptedCmp2         2239 non-null   int64
 25  Complain             2239 non-null   int64
 26  Country              2239 non-null   object
dtypes: int64(22), object(5)
memory usage: 472.4+ KB
```

In [36]:

```
campaign.describe()
```

Out[36]:

	ID	Year_Birth	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProducts	MntSweetProducts	...	Nu
count	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	2239.000000	...	
mean	5590.444841	1968.802144	0.443948	0.506476	49.121036	304.067441	26.307727	167.016525	37.538633	27.074587	...	
std	3246.372471	11.985494	0.538390	0.544555	28.963662	336.614830	39.781468	225.743829	54.637617	41.286043	...	
min	0.000000	1893.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	2827.500000	1959.000000	0.000000	0.000000	24.000000	24.000000	1.000000	16.000000	3.000000	1.000000	...	
50%	5455.000000	1970.000000	0.000000	0.000000	49.000000	174.000000	8.000000	67.000000	12.000000	8.000000	...	
75%	8423.500000	1977.000000	1.000000	1.000000	74.000000	504.500000	33.000000	232.000000	50.000000	33.000000	...	
max	11191.000000	1996.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	263.000000	...	

8 rows × 22 columns

In [37]:

In [37]:

```
campaign.columns
```

Out[37]:

```
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Country'],
      dtype='object')
```

Is income of customers dependent on their education ?

In [38]:

```
c1 = campaign.copy()
c1['Dt_Customer'] = pd.to_datetime(c1['Dt_Customer'])
c1['ID'] = c1['ID'].apply(str)
c1['Income'] = c1['Income'].str.replace('$', '')
c1['Income'] = c1['Income'].str.replace(',', '')
c1['Income'] = pd.to_numeric(c1['Income'], errors='coerce').fillna(0)
c1['Income'] = c1['Income'].round(0).astype(int)
df_EI = c1[['Education','Income']]
a = df_EI[df_EI['Education']=='Graduation']
b = df_EI[df_EI['Education']=='PhD']
c = df_EI[df_EI['Education']=='2n Cycle']
d = df_EI[df_EI['Education']=='Master']
e = df_EI[df_EI['Education']=='Basic']
n1 = a['Income'].sample(100)
n2 = b['Income'].sample(100)
n3 = c['Income'].sample(100)
n4 = d['Income'].sample(100)
n5 = e['Income'].sample(50)
alpha = 0.05
Ho = 'There is no significant effect of Education on Income'
Ha = 'There is significant effect of Education on Income'
statistic,pvalue = f_oneway(n1,n2,n3,n4,n5)
if pvalue < alpha :
  print(pvalue)
```

```
  print('Reject Null Hypothesis')
  print(Ha)
else:
  print(pvalue)
  print('Fail to Reject Null Hypothesis')
  print(Ho)
```

```
6.364152226657658e-23
Reject Null Hypothesis
There is significant effect of Education on Income
```

```
<ipython-input-38-47c80646494f>:2: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, please specify a format.
  c1['Dt_Customer'] = pd.to_datetime(c1['Dt_Customer'])
```

Test for Distribution

In [39]:

```
Ho = 'The data follows Normal Distribution'
Ha = 'The data does not follows Normal Distribution'
alpha = 0.05
n1_stat, n1_pval = shapiro(n1)
n2_stat, n2_pval = shapiro(n2)
n3_stat, n3_pval = shapiro(n3)
n4_stat, n4_pval = shapiro(n4)
n5_stat, n5_pval = shapiro(n5)
if pvalue < alpha :
  print('Reject Null Hypotheses')
  print("Shapiro-Wilk Test Results:")
  print("Education Group 1:", n1_stat, n1_pval)
  print("Education Group 2:", n2_stat, n2_pval)
  print("Education Group 3:", n3_stat, n3_pval)
  print("Education Group 4:", n4_stat, n4_pval)
  print("Education Group 5:", n5_stat, n5_pval)
else :
  print('Fail to Reject Null Hypotheses')
  print("Shapiro-Wilk Test Results:")
  print("Education Group 1:", n1_stat, n1_pval)
  print("Education Group 2:", n2_stat, n2_pval)
  print("Education Group 3:", n3_stat, n3_pval)
  print("Education Group 4:", n4_stat, n4_pval)
  print("Education Group 5:", n5_stat, n5_pval)
```

```
Reject Null Hypotheses
```

```
Shapiro-Wilk Test Results:
Education Group 1: 0.9599016849372809 0.003956821625062427
Education Group 2: 0.9087516051661321 3.7358017611627106e-06
Education Group 3: 0.9473413026293287 0.0005566050625773607
Education Group 4: 0.9830978809665438 0.22986527342138935
Education Group 5: 0.955174758175495 0.05586196266045785
```

Test for Variance

In [40]:

```
alpha = 0.05
Ho = 'The variances of all groups are equal'
Ha = 'At least one groups variance is different from the others'
statistic,pvalue = levene(n1,n2,n3,n4,n5)
if pvalue < alpha :
  print(pvalue)
  print('Reject Null Hypothesis')
  print(Ha)
else:
  print(pvalue)
  print('Fail to Reject Null Hypothesis')
  print(Ho)
```

```
1.2002232963493167e-14
Reject Null Hypothesis
At least one groups variance is different from the others
```

Since the assumptions for anova tests are failed conducting kruskal walis test

In [41]:

```
alpha = 0.05
Ho = 'There is no significant effect of Education on Income'
Ha = 'There is significant effect of Education on Income'
statistic,pvalue = kruskal(n1,n2,n3,n4,n5)
if pvalue < alpha :
  print(pvalue)
  print('Reject Null Hypothesis')
  print(Ha)
else:
  print(pvalue)
  print('Fail to Reject Null Hypothesis')
```

```
print(Ho)
```

```
5.098131925928622e-22
Reject Null Hypothesis
There is significant effect of Education on Income
```

In [42]:

```
c1.groupby('Education')['Income'].mean().reset_index()
```

Out[42]:

	Education	Income
0	2n Cycle	46929.251232
1	Basic	20306.259259
2	Graduation	51660.098579
3	Master	52202.432432
4	PhD	55567.687243

Insights and Recommendations:

- **From the above tests conducted we can assume that there is a significant effect of Education of Customers on their Income**
- **We can categorise the markeing mails accordingly based on the category of their education**
- **Further tests can be conducted on their spending nature and their value added to the business**

Do higher income people spend more (take in account spending in all categories together)?

In [43]:

```
c1['Total_Spent'] = c1['MntWines']+c1['MntSweetProducts']+c1['MntMeatProducts']+c1['MntGoldProds']+c1['MntFruits']+c1['MntFishProducts']
spent = c1[['ID','Education','Marital_Status','Income','Total_Spent']]
spent.sort_values(by='Income',ascending=False).head()
```

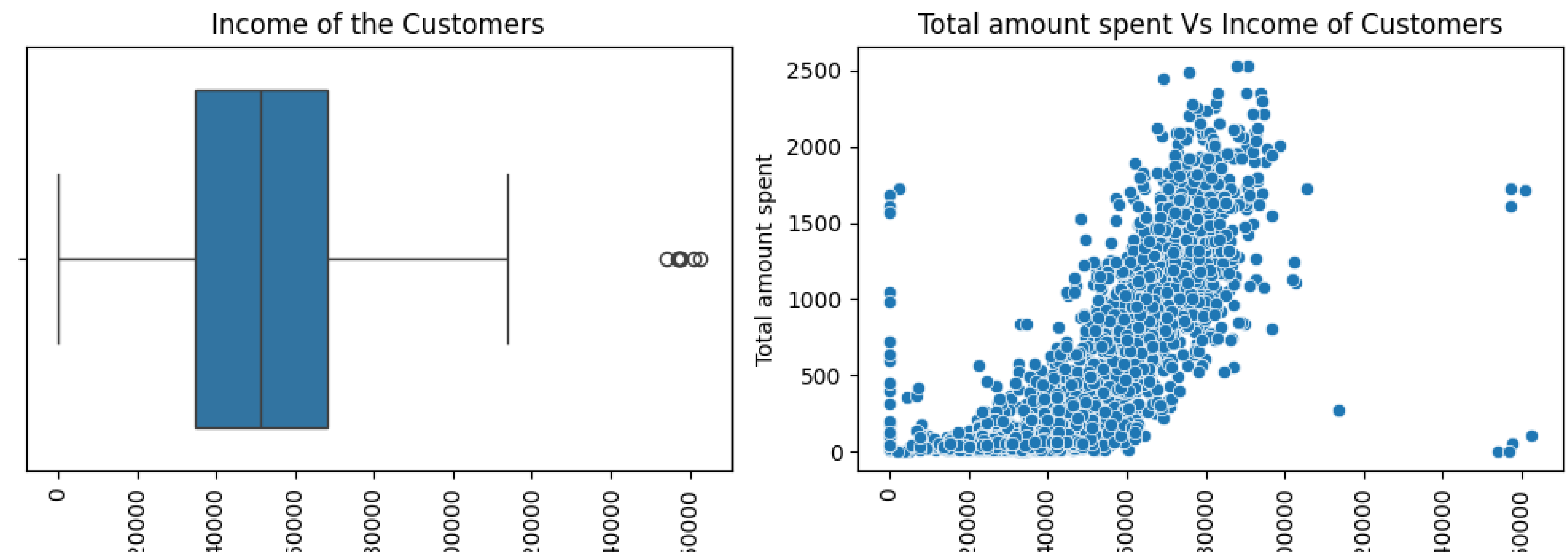
Out[43]:

	ID	Education	Marital_Status	Income	Total_Spent

	ID	Education	Marital_Status	Income	Total_Spent
730	1503	PhD	Together	162397	107
497	1501	PhD	Married	160803	1717
852	5336	Master	Together	157733	59
2203	8475	PhD	Married	157243	1608
325	4931	Graduation	Together	157146	1730

In [44]:

```
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
sns.boxplot(data=spent,x='Income')
plt.title('Income of the Customers')
plt.xticks(rotation=90)
plt.subplot(1,2,2)
sns.scatterplot(data=spent,x='Income',y='Total_Spent')
plt.title('Total amount spent Vs Income of Customers')
plt.xlabel('Income of Customers')
plt.ylabel('Total amount spent')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



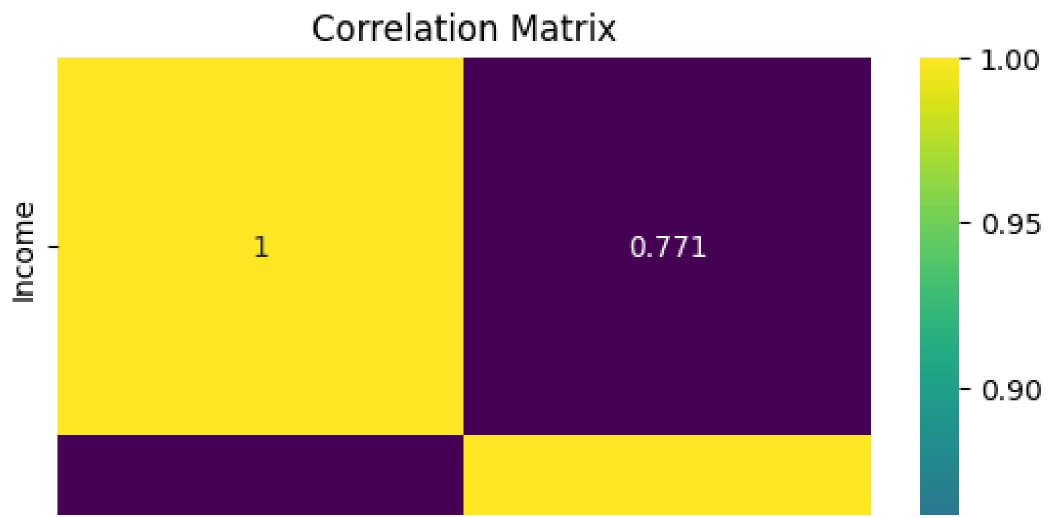
In [45]:

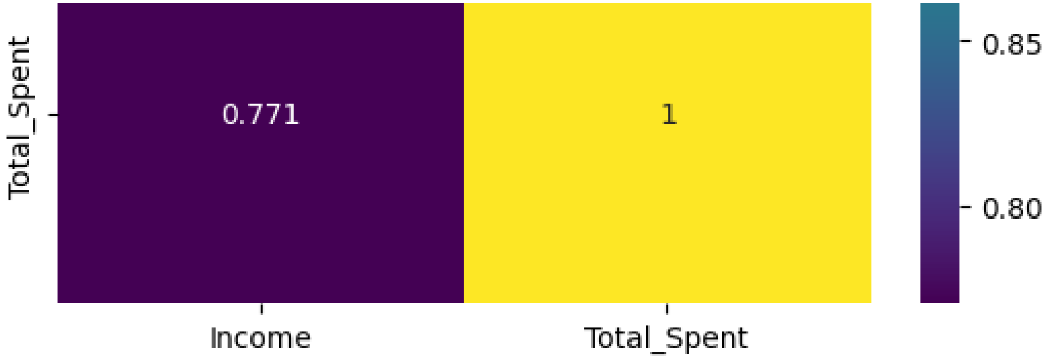
```
alpha = 0.05
Ho = 'There is no significant relation between Income and Total ampount spent'
Ha = 'There is significant relation between Income and Total ampount spent'
#statistic, pvalue = pearsonr(spent['Income'],spent['Total_Spent'])
correlation_coefficient, p_value = pearsonr(spent['Income'], spent['Total_Spent'])
if pvalue < alpha :
  print(pvalue)
  print('Reject Null Hypothesis')
  print(Ha)
else:
  print(pvalue)
  print('Fail to Reject Null Hypothesis')
  print(Ho)
```

```
5.098131925928622e-22
Reject Null Hypothesis
There is significant relation between Income and Total ampount spent
```

In [46]:

```
correlation_matrix = spent[['Income','Total_Spent']].corr()
sns.heatmap(correlation_matrix,annot=True,cmap='viridis',fmt='.3g')
plt.title('Correlation Matrix')
plt.show()
```





Insights and Recommendations:

- **From the above tests conducted it can be assumed that there is a significant relation between income of the customer and total amount spent**
- **We can observe strong positive correlation between income of the customer and total amount spent**
- **Observed some outliers with high Income customers and low amount spent by them, further tests can be conducted on the demographics for further analyses like location etc**

In [47]:

```
spent.sort_values(by='Total_Spent',ascending=False).head(10)
```

Out[47]:

	ID	Education	Marital_Status	Income	Total_Spent
671	5350	Master	Single	90638	2525
670	5735	Master	Single	90638	2525
1403	1763	Graduation	Together	87679	2524
1025	4580	Graduation	Married	75759	2486
1863	4475	PhD	Married	69098	2440
606	5453	Master	Married	90226	2352
376	10133	Graduation	Single	93790	2349
1810	9010	Master	Married	83151	2346
1408	6024	Graduation	Together	94384	2302
1407	5386	Graduation	Together	94384	2302

Do couples spend more or less money on wine than people living alone (set 'Married','Together':'In couple' and 'Divorced','Single','Absurd','Widow','YOLO':'Alone')

In [48]:

```
campaign['Marital_Status'].value_counts()
```

Out[48]:

	count
Marital_Status	
Married	864
Together	579
Single	480
Divorced	232
Widow	77
Alone	3
YOLO	2
Absurd	2

dtype: int64

In [49]:

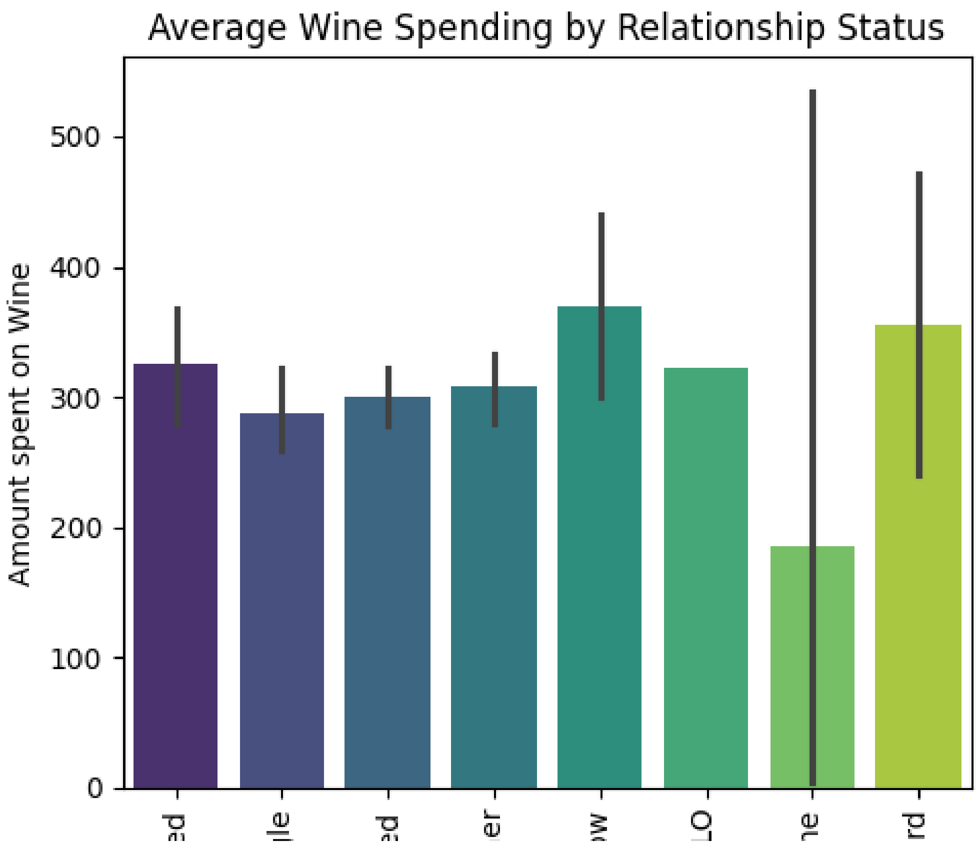
```
df1 = c1.copy()
z1 = df1[df1['Marital_Status'].isin(['Married','Together'])]
z2 = df1[~df1['Marital_Status'].isin(['Married','Together'])]
Couple = z1[['ID','Marital_Status','MntWines']]
Alone = z2[['ID','Marital_Status','MntWines']]
alpha = 0.05
Ho = 'Total amount spent on wines by couples and people living alone have no significant differnece'
Ha = 'Total amount spent by wines by couples and people living alone are significantly different'
statistic, pvalue = ttest_ind(Couple['MntWines'].sample(500),Alone['MntWines'].sample(500))
if pvalue < alpha :
  print(pvalue)
  print('Reject Null Hypothesis')
```

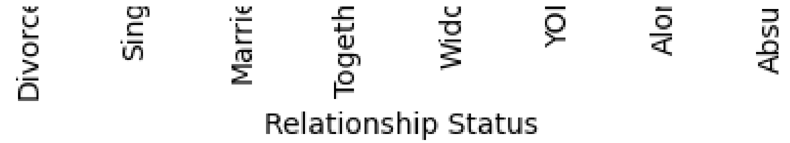
```
    print(Ha)
else:
    print(pvalue)
    print('Fail to Reject Null Hypothesis')
    print(Ho)
```

```
0.5011816522565158
Fail to Reject Null Hypothesis
Total amount spent on wines by couples and people living alone have no significant differnece
```

In [50]:

```
plt.figure(figsize=(5,5))
sns.barplot(data=df1,x='Marital_Status',y='MntWines',palette='viridis',hue='Marital_Status')
plt.title('Average Wine Spending by Relationship Status')
plt.xlabel('Relationship Status')
plt.ylabel('Amount spent on Wine')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



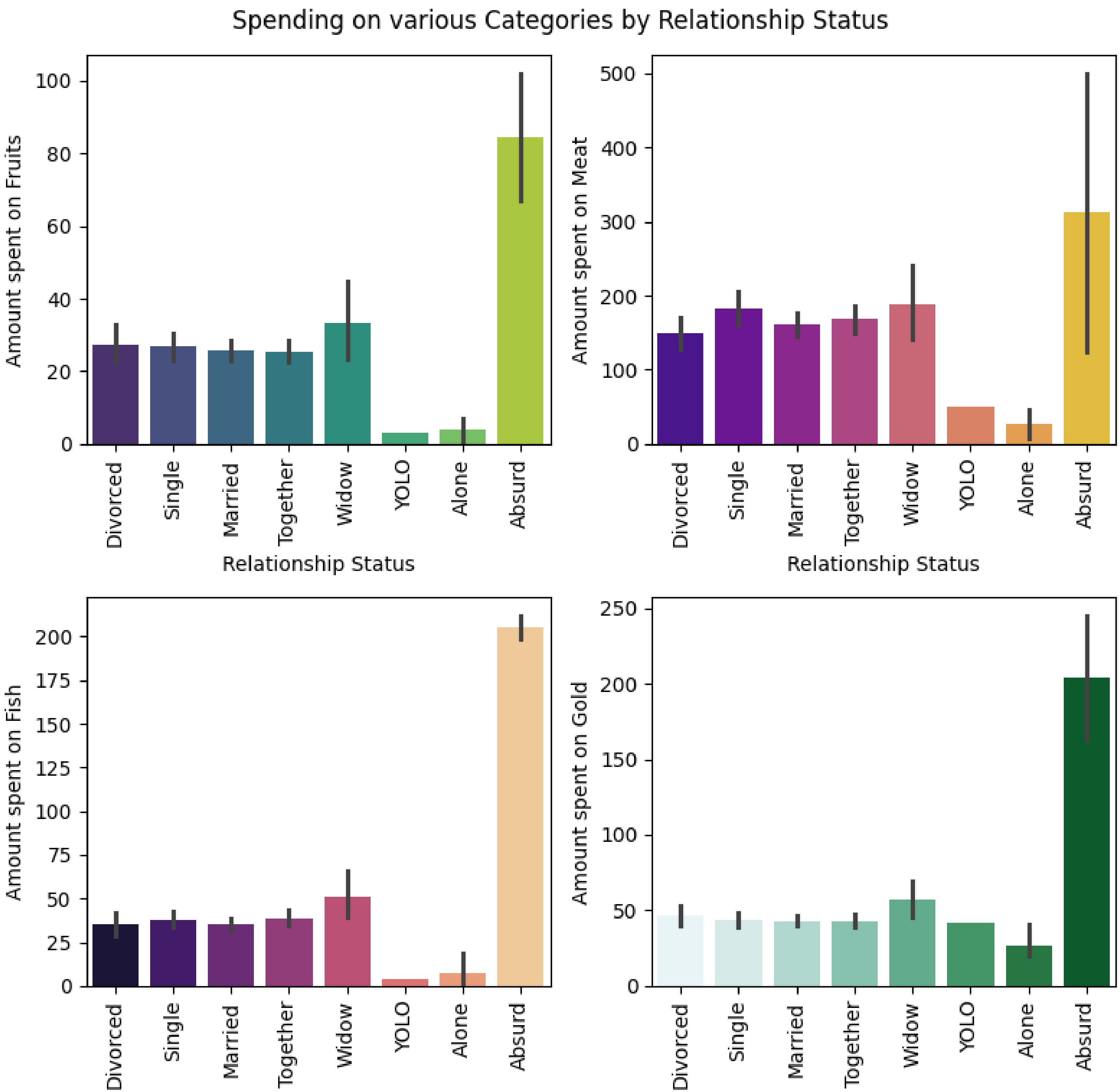


Insights and Recommendations:

- **From the above tests conducted it can be deduced that there is no significant difference in the total amount spent by the married couple and people alone**
- **The mean amount spent on the wines is almost same for each category**
- **Further tests need to be conducted on the preference of the amount spent on various other categories**

In [51]:

```
plt.figure(figsize=(8,8))
plt.subplot(2,2,1)
sns.barplot(data=df1,x='Marital_Status',y='MntFruits',palette='viridis',hue='Marital_Status')
#plt.title('Average Spending by Relationship Status')
plt.xlabel('Relationship Status')
plt.ylabel('Amount spent on Fruits')
plt.xticks(rotation=90)
plt.subplot(2,2,2)
sns.barplot(data=df1,x='Marital_Status',y='MntMeatProducts',palette='plasma',hue='Marital_Status')
#plt.title('Average Spending by Relationship Status')
plt.xlabel('Relationship Status')
plt.ylabel('Amount spent on Meat')
plt.xticks(rotation=90)
plt.subplot(2,2,3)
sns.barplot(data=df1,x='Marital_Status',y='MntFishProducts',palette='magma',hue='Marital_Status')
#plt.title('Average Spending by Relationship Status')
plt.xlabel('Relationship Status')
plt.ylabel('Amount spent on Fish')
plt.xticks(rotation=90)
plt.subplot(2,2,4)
sns.barplot(data=df1,x='Marital_Status',y='MntGoldProds',palette='BuGn',hue='Marital_Status')
#plt.title('Average  Spending by Relationship Status')
plt.xlabel('Relationship Status')
plt.ylabel('Amount spent on Gold')
plt.xticks(rotation=90)
plt.suptitle('Spending on various Categories by Relationship Status')
plt.tight_layout()
plt.show()
```


Spending on various Categories by Relationship Status
Amount spent on Fruits
Amount spent on Meat
Amount spent on Fish
Amount spent on Gold
Relationship Status
Relationship Status
Divorced
Single
Married
Together
Widow
YOLO
Alone
Absurd
0
20
40
60
80
100
0
100
200
300
400
500
0
25
50
75
100
125
150
175
200
0
50
100
150
200
250

Relationship Status Relationship Status

In [52]:

```
df1.head()
```

Out[52]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumStorePurchases	NumWebVisitsMonth	Ac
0	1826	1970	Graduation	Divorced	84835	0	0	2014-06-16	0	189	...	6	1	
1	1	1961	Graduation	Single	57091	0	0	2014-06-15	0	464	...	7	5	
2	10476	1958	Graduation	Married	67267	0	1	2014-05-13	0	134	...	5	2	
3	1386	1967	Graduation	Together	32474	1	1	2014-05-11	0	10	...	2	7	
4	5371	1989	Graduation	Single	21474	1	0	2014-04-08	0	6	...	2	7	

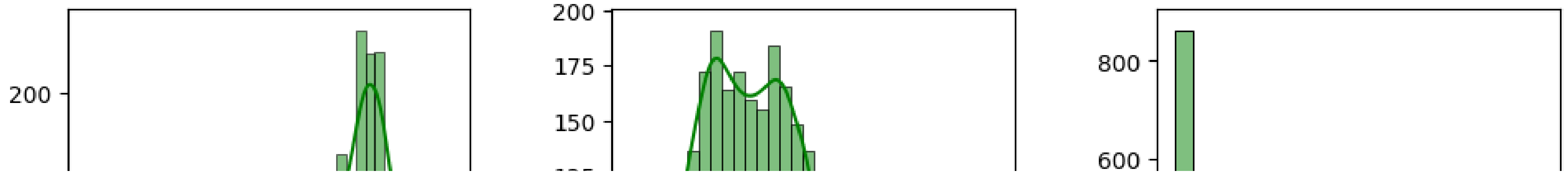
5 rows × 28 columns

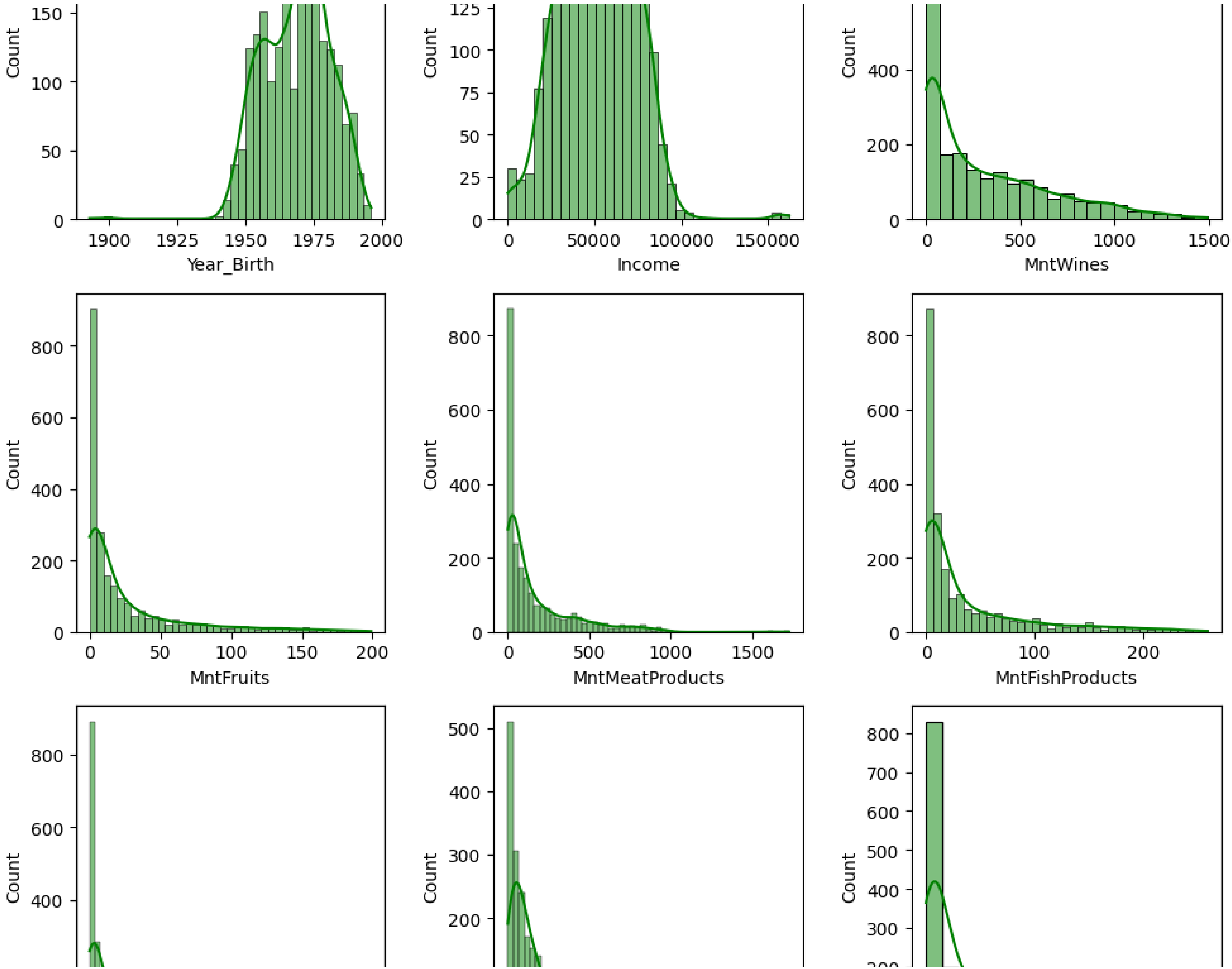
Exploratory Analyses

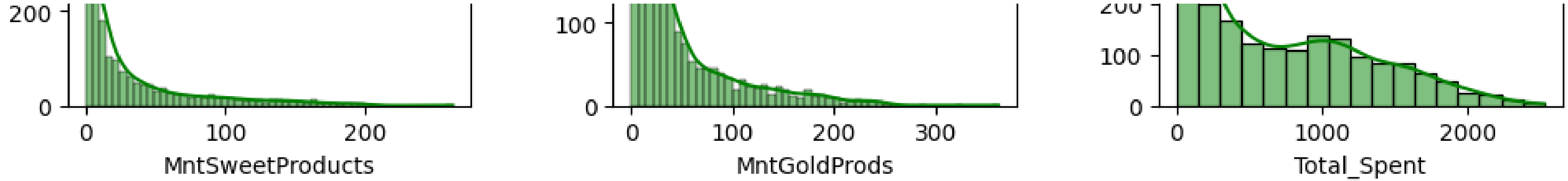
univariate analyses

In [53]:

```
numerical = ['Year_Birth','Income','MntWines', 'MntFruits','MntMeatProducts',
             'MntFishProducts', 'MntSweetProducts','MntGoldProds','Total_Spent']
fig, axes = plt.subplots(3,3,figsize=(10,10))
for i, ax in enumerate(axes.flatten()):
  sns.histplot(data=df1[numerical[i]],ax=ax,kde=True,color='green')
plt.tight_layout()
plt.show()
```



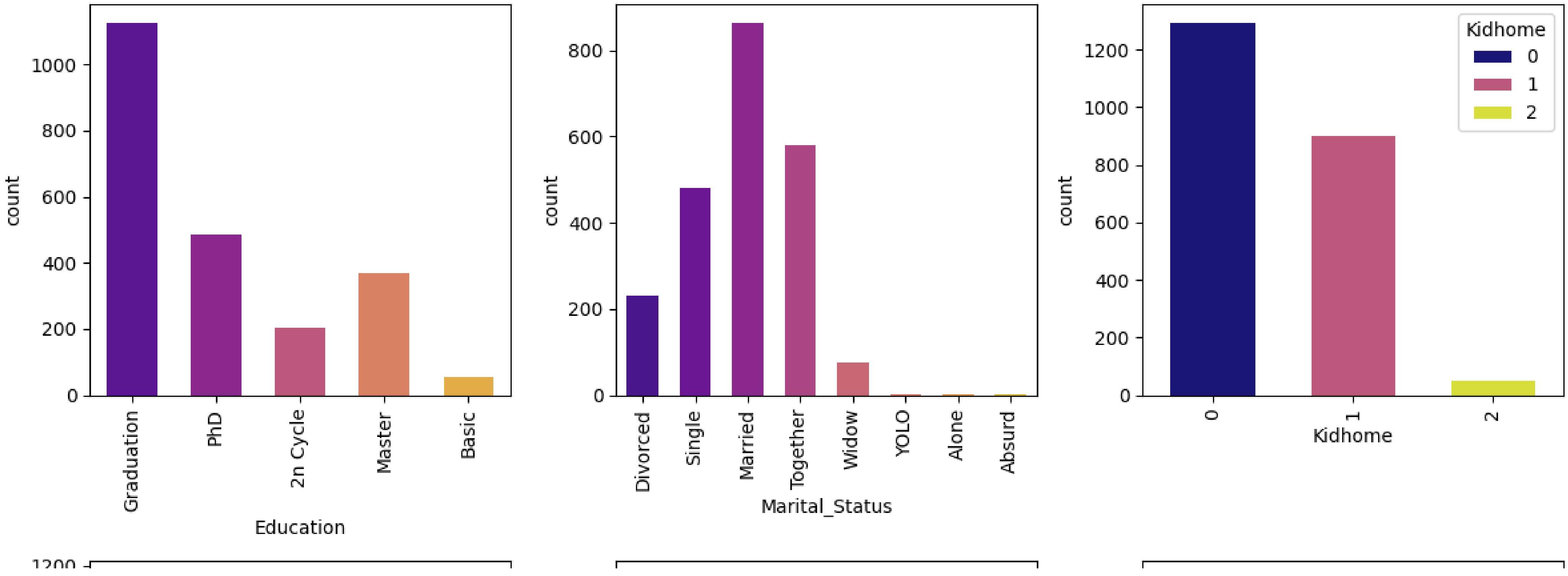

Count
150
100
50
0
1900
1925
1950
1975
2000
Year_Birth
Count
125
100
75
50
25
0
0
50000
100000
150000
Income
Count
400
200
0
0
500
1000
1500
MntWines
Count
800
600
400
200
0
0
50
100
150
200
MntFruits
Count
800
600
400
200
0
0
500
1000
1500
MntMeatProducts
Count
800
600
400
200
0
0
100
200
MntFishProducts
Count
800
600
400
Count
500
400
300
200
Count
800
700
600
500
400
300



Bi-variate Analyses

In [54]:

```
categorical=['Education', 'Marital_Status','Kidhome',
             'Teenhome','Complain', 'Country']
fig, axes = plt.subplots(2, 3, figsize=(12,8))
axes = axes.flatten()
for i, col in enumerate(categorical):
  sns.countplot(x=df1[col], ax=axes[i],hue=df1[col],palette='plasma',width=0.6)
  axes[i].set_xticks(axes[i].get_xticks())
  axes[i].set_xticklabels(axes[i].get_xticklabels(), rotation=90)
plt.tight_layout()
plt.show()
```



Are people with lower income are more attracted towards campaign or simply put accept more campaigns. (create two income brackets one below median , other above median income and create a column which tells if they have ever accepted any campaign)

In [55]:

```
df2 = c1.copy()
m = df2['Income'].median()
df2['Income_Category'] = df2['Income'].apply(lambda x: 'High' if x>m else 'Low')
df2.head()
```

Out[55]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth	AcceptedCmp3	Accepte
0	1826	1970	Graduation	Divorced	84835	0	0	2014-06-16	0	189	...	1	0	
1	1	1961	Graduation	Single	57091	0	0	2014-06-15	0	464	...	5	0	
2	10476	1958	Graduation	Married	67267	0	1	2014-05-13	0	134	...	2	0	
3	1386	1967	Graduation	Together	32474	1	1	2014-05-11	0	10	...	7	0	
4	5371	1989	Graduation	Single	21474	1	0	2014-04-08	0	6	...	7	1	

5 rows × 29 columns

In [56]:

```
df2['Campaign_Accepted'] = df2[['AcceptedCmp1','AcceptedCmp2','AcceptedCmp3','AcceptedCmp4','AcceptedCmp5']].apply(lambda
x: 'Yes' if any(x == 1) else 'No',axis =1)
df_category = df2[['Income_Category','Campaign_Accepted']]
contingency_table = pd.crosstab(df_category['Income_Category'],df_category['Campaign_Accepted'])
contingency_table
```

Out[56]:

Campaign_Accepted	No	Yes
Income_Category		
High	774	345
Low	1002	118

In [57]:

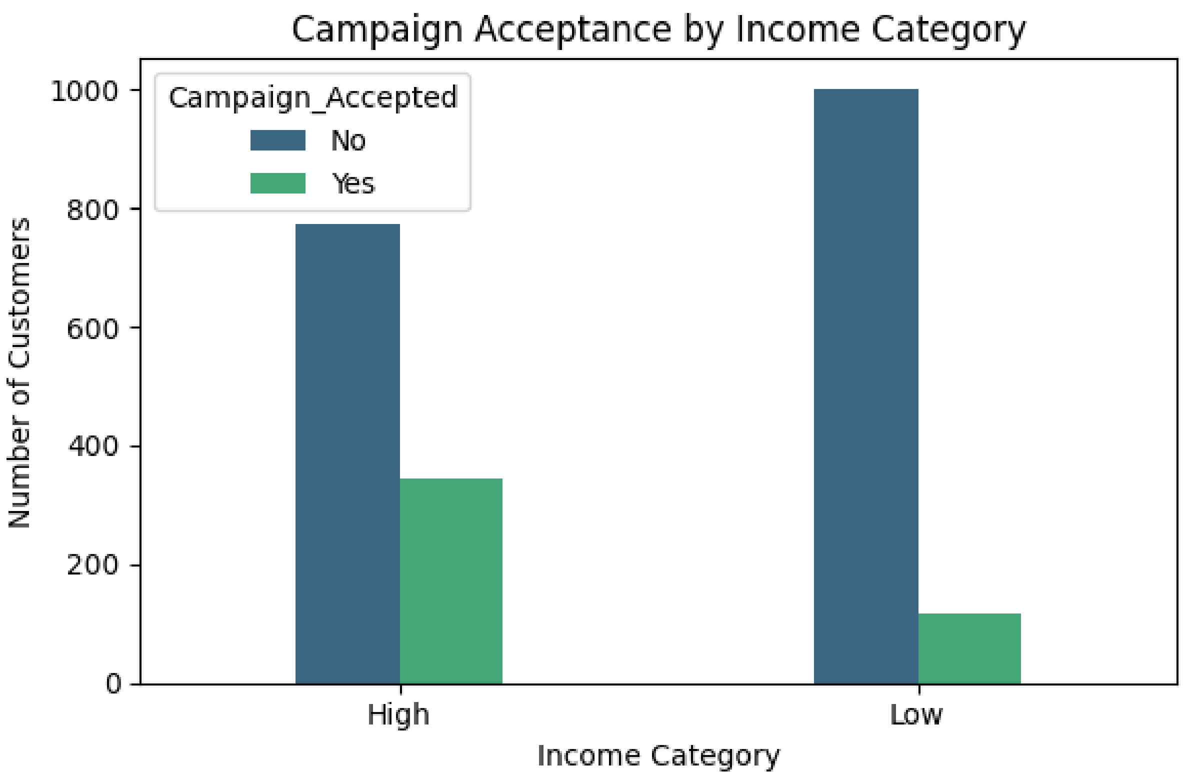
```
alpha  = 0.05
Ho = 'There is no significant difference in acceptance between two income groups'
Ha = 'There is a significant difference in acceptance between two income groups'
statistic, pvalue, dof, expected_freq = chi2_contingency(contingency_table)
if pvalue < alpha :
  print(pvalue)
  print('Reject Null Hypothesis')
  print(Ha)
else:
  print(pvalue)
  print('Fail to Reject Null Hypothesis')
  print(Ho)
```

```
3.7320719296764283e-32
Reject Null Hypothesis
There is a significant difference in acceptance between two income groups
```

In [58]:

```
plt.figure(figsize=(6,4))
sns.countplot(data=df_category,x='Income_Category',hue='Campaign_Accepted',palette='viridis',width=0.4)
plt.title('Campaign Acceptance by Income Category')
```

```
plt.xlabel('Income Category')
plt.ylabel('Number of Customers')
plt.tight_layout()
plt.show()
```



Insights and Recommendations

- **From the above tests it can be concluded that there is a significant differnce in the campaign acceptance of customers and their Income level**
- **Customers with high income levels tend to accept the campaigns much higher than the customers with low income levels**

Analyses on shopping Dataset

In [59]:

```
shopping.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
```

```
Data columns (total 18 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Administrative           12330 non-null  int64  
 1   Administrative_Duration  12330 non-null  float64
 2   Informational            12330 non-null  int64  
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  int64  
 5   ProductRelated_Duration  12330 non-null  float64
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object 
 11  OperatingSystems         12330 non-null  int64  
 12  Browser                  12330 non-null  int64  
 13  Region                   12330 non-null  int64  
 14  TrafficType              12330 non-null  int64  
 15  VisitorType              12330 non-null  object 
 16  Weekend                  12330 non-null  bool   
 17  Revenue                  12330 non-null  bool   
dtypes: bool(2), float64(7), int64(7), object(2)
memory usage: 1.5+ MB
```

Exploratory Analyses

In [60]:

```
Numerical =
['Administrative','Informational','ProductRelated','BounceRates','ExitRates','PageValues','Administrative_Duration','Info
rmational_Duration',
             'ProductRelated_Duration']
```

In [61]:

```
shopping[Numerical].describe(include='all')
```

Out[61]:

	Administrative	Informational	ProductRelated	BounceRates	ExitRates	PageValues	Administrative_Duration	Informational_Duration	ProductRelated_Dura
count	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000000	12330.000
mean	2.315166	0.503569	31.731468	0.022191	0.043073	5.889258	80.818611	34.472398	1194.746

std	Administrative	Informational	ProductRelated	BounceRates	ExitRates	PageValues	Administrative_Duration	Informational_Duration	ProductRelated_Dura
std	3.321784	1.270156	44.475503	0.048488	0.048597	18.568437	176.779107	140.749294	1913.66
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	0.000000	0.000000	7.000000	0.000000	0.014286	0.000000	0.000000	0.000000	184.137
50%	1.000000	0.000000	18.000000	0.003112	0.025156	0.000000	7.500000	0.000000	598.936
75%	4.000000	0.000000	38.000000	0.016813	0.050000	0.000000	93.256250	0.000000	1464.157
max	27.000000	24.000000	705.000000	0.200000	0.200000	361.763742	3398.750000	2549.375000	63973.522

In [62]:

```
def z_outlier(df,columns,threshold=3):
  for i in columns:
    z_scores = stats.zscore(df[i])
    df = df[(z_scores< threshold) & (z_scores > -threshold)]
  return df
```

In [63]:

```
df_cleaned = z_outlier(shopping,Numerical)
df_cleaned.head()
```

```
<ipython-input-62-1dceae200844>:3: DeprecationWarning: Please import `zscore` from the `scipy.stats` namespace; the `sci
py.stats.stats` namespace is deprecated and will be removed in SciPy 2.0.0.
  z_scores = stats.zscore(df[i])
```

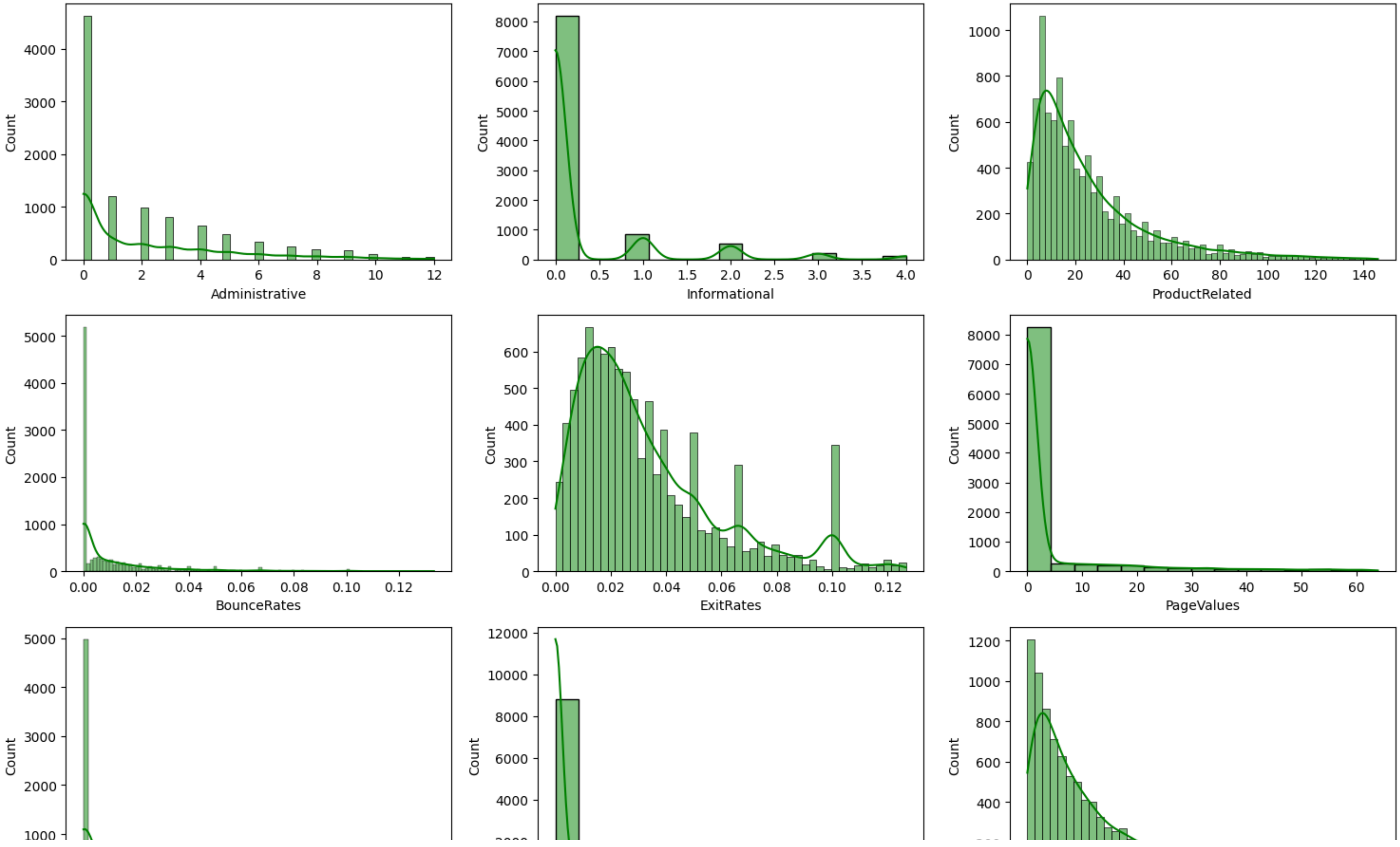
Out[63]:

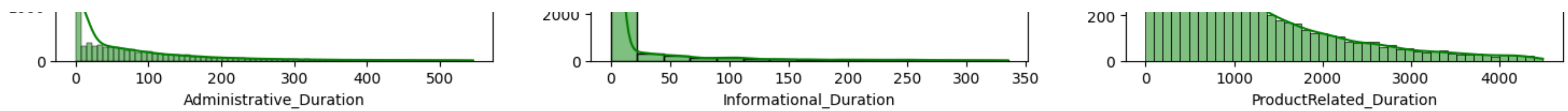
	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	Spe
1	0	0.0	0	0.0	2	64.000000	0.000000	0.100000	0.0	
4	0	0.0	0	0.0	10	627.500000	0.020000	0.050000	0.0	
5	0	0.0	0	0.0	19	154.216667	0.015789	0.024561	0.0	
8	0	0.0	0	0.0	2	37.000000	0.000000	0.100000	0.0	
9	0	0.0	0	0.0	3	738.000000	0.000000	0.022222	0.0	

In [64]:

```
Numerical =
```

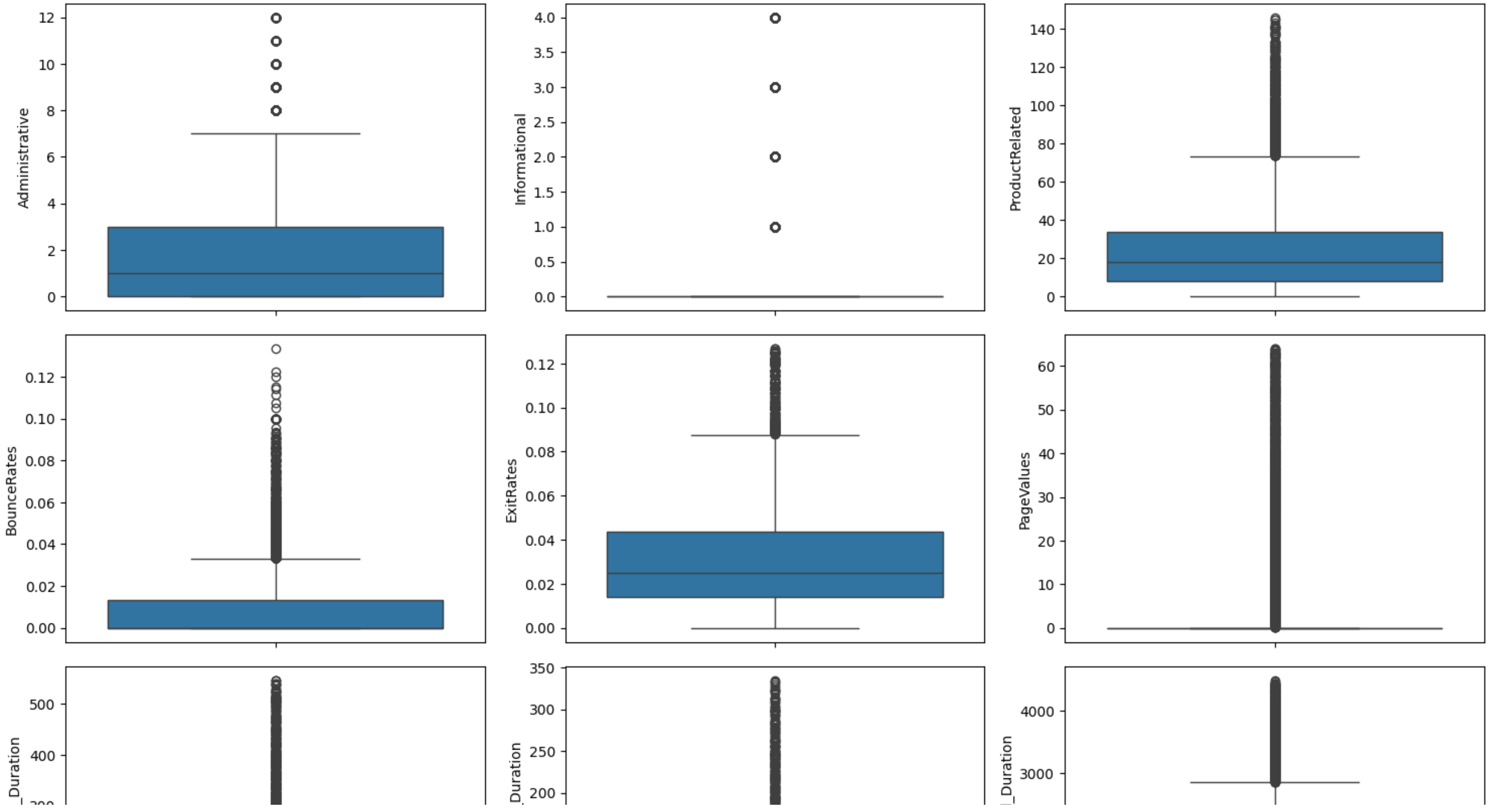
```
Numerical =
['Administrative','Informational','ProductRelated','BounceRates','ExitRates','PageValues','Administrative_Duration','Info
rmational_Duration',
             'ProductRelated_Duration']
fig, axes = plt.subplots(3, 3, figsize=(15, 10))
for i, ax in enumerate(axes.flatten()):
  sns.histplot(df_cleaned[Numerical[i]],ax=ax,kde=True,color='green')
plt.tight_layout()
plt.show()
```

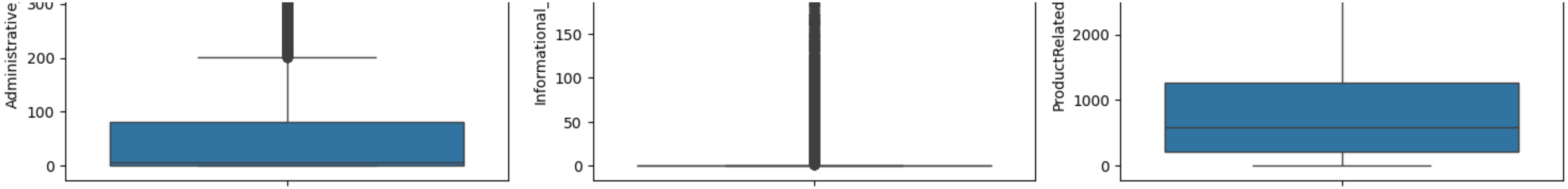




In [65]:

```
fig, axes = plt.subplots(3, 3, figsize=(15, 10))
for i, ax in enumerate(axes.flatten()):
  sns.boxplot(df_cleaned[Numerical[i]],ax=ax)
plt.tight_layout()
plt.show()
```





In [66]:

```
Categorical = ['SpecialDay', 'Month','OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
        'Weekend', 'Revenue']
```

In [67]:

```
Categorical = ['SpecialDay', 'Month','OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
        'Weekend', 'Revenue']
for i in Categorical:
    frequency_counts = shopping[i].value_counts()
    print(f"\nFrequency counts for {i}:\n")
    print(frequency_counts)
```

```
Frequency counts for SpecialDay:

SpecialDay
0.0    11079
0.6      351
0.8      325
0.4      243
0.2      178
1.0      154
Name: count, dtype: int64

Frequency counts for Month:

Month
May     3364
Nov     2998
Mar     1907
Dec     1727
Oct      549
Sep      448
Aug      433
Jul      432
June     288
```

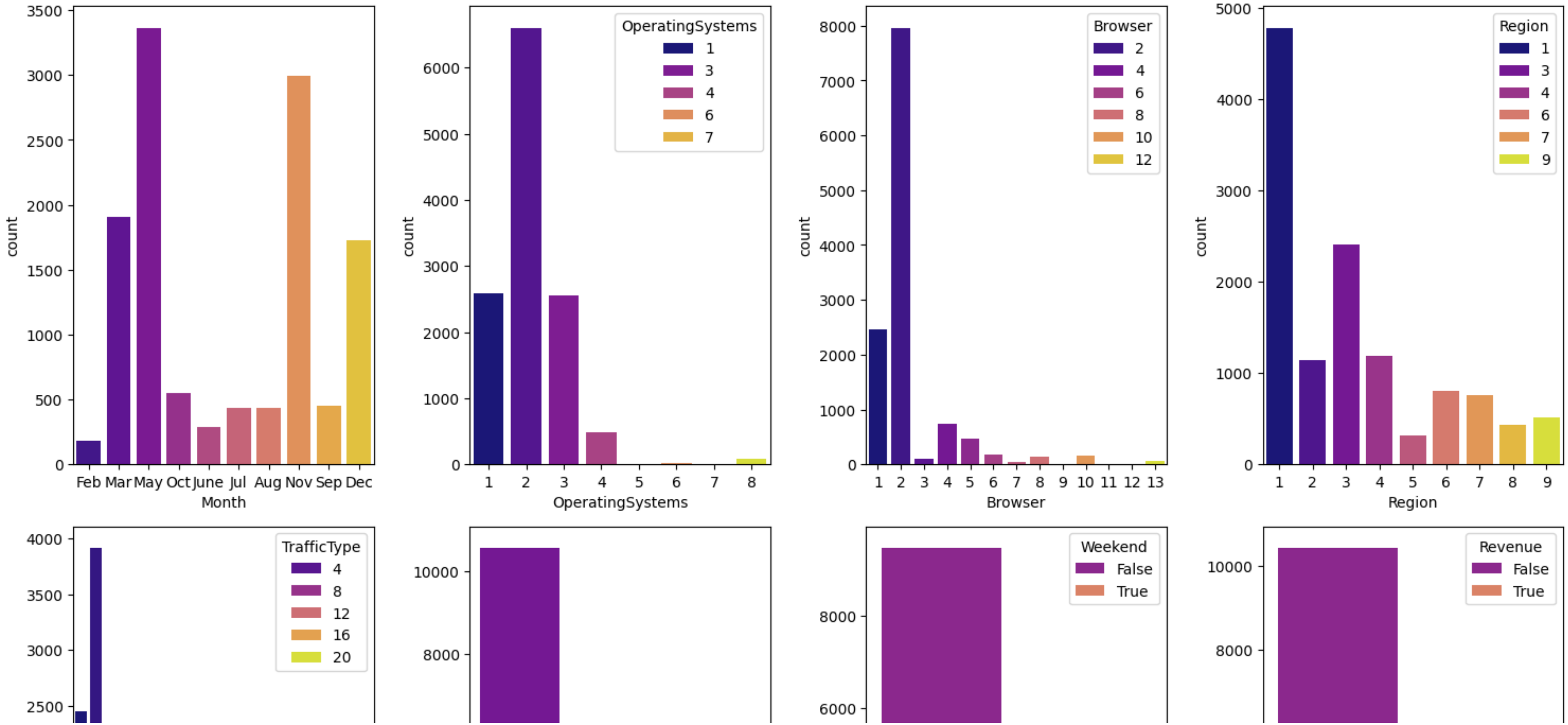
```
Feb       184
Name: count, dtype: int64

Frequency counts for OperatingSystems:

OperatingSystems
2    6601
1    2585
3    2555
4     478
8      79
6      19
7       7
5       6
Name: count, dtype: int64

Frequency counts for Browser:

Browser
2     7961
1     2462
4      736
5      467
6      174
10     163
8      135
3      105
13      61
7       49
12      10
11       6
9        1
Name: count, dtype: int64

Frequency counts for Region:

Region
1    4780
3    2403
4    1182
2    1136
6     805
7     761
9     511
8     434
```

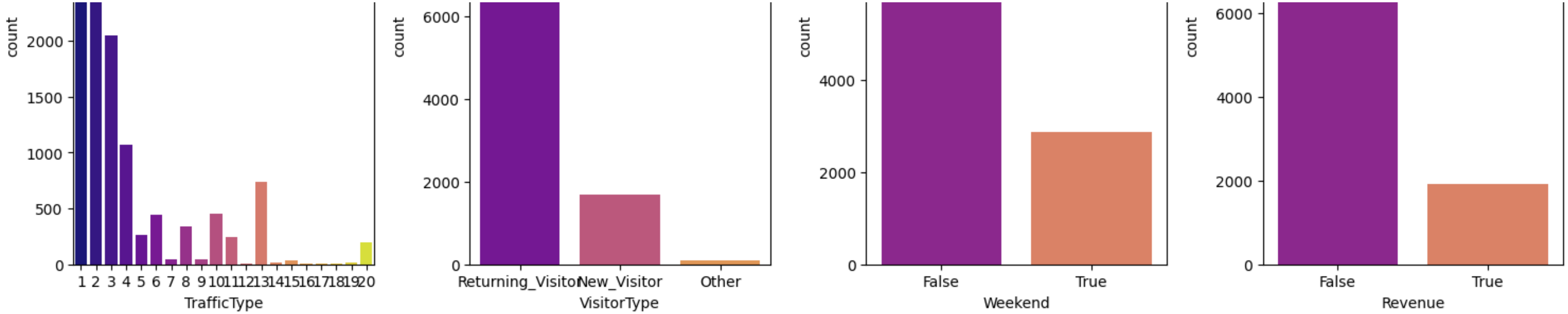
```
5     318
Name: count, dtype: int64

Frequency counts for TrafficType:

TrafficType
2     3913
1     2451
3     2052
4     1069
13     738
10     450
6      444
8      343
5      260
11     247
20     198
9       42
7       40
15      38
19      17
14      13
18      10
16       3
12       1
17       1
Name: count, dtype: int64

Frequency counts for VisitorType:

VisitorType
Returning_Visitor    10551
New_Visitor           1694
Other                   85
Name: count, dtype: int64

Frequency counts for Weekend:

Weekend
False    9462
True     2868
Name: count, dtype: int64

Frequency counts for Revenue:
```

```
Revenue
False    10422
True      1908
Name: count, dtype: int64
```

In [68]:

```
Categorical = ['Month','OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
       'Weekend', 'Revenue']
fig, axes = plt.subplots(2, 4, figsize=(15, 10))
axes = axes.flatten()
for i, col in enumerate(Categorical):
    sns.countplot(x=shopping[col], ax=axes[i],hue=shopping[col],palette='plasma')
# Adjust the layout for better spacing between plots
plt.tight_layout()
plt.show()
```





correlations between numerical features to identify potential relationships.

In [70]:

```
Numerical = ['Administrative','Informational','ProductRelated','BounceRates','ExitRates','PageValues','Administrative_Duration','Informational_Duration',
             'ProductRelated_Duration']
shopping[Numerical].corr()
```

Out[70]:

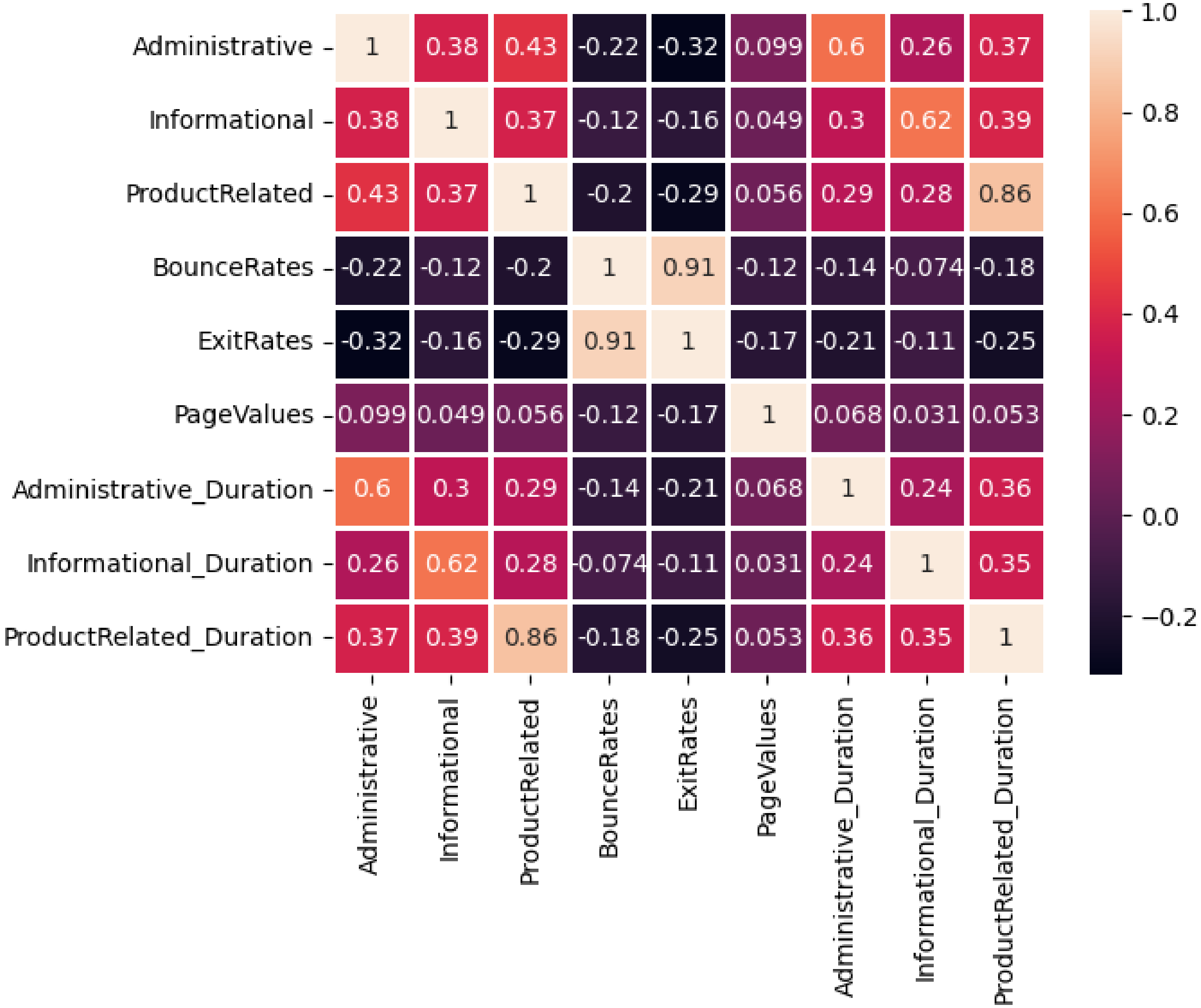
	Administrative	Informational	ProductRelated	BounceRates	ExitRates	PageValues	Administrative_Duration	Informational_Duration	Product
Administrative	1.000000	0.376850	0.431119	-0.223563	-0.316483	0.098990	0.601583	0.255848	
Informational	0.376850	1.000000	0.374164	-0.116114	-0.163666	0.048632	0.302710	0.618955	
ProductRelated	0.431119	0.374164	1.000000	-0.204578	-0.292526	0.056282	0.289087	0.280046	
BounceRates	-0.223563	-0.116114	-0.204578	1.000000	0.913004	-0.119386	-0.144170	-0.074067	
ExitRates	-0.316483	-0.163666	-0.292526	0.913004	1.000000	-0.174498	-0.205798	-0.105276	
PageValues	0.098990	0.048632	0.056282	-0.119386	-0.174498	1.000000	0.067608	0.030861	
Administrative_Duration	0.601583	0.302710	0.289087	-0.144170	-0.205798	0.067608	1.000000	0.238031	
Informational_Duration	0.255848	0.618955	0.280046	-0.074067	-0.105276	0.030861	0.238031	1.000000	
ProductRelated_Duration	0.373939	0.387505	0.860927	-0.184541	-0.251984	0.052823	0.355422	0.347364	

In [71]:

```
x = shopping[Numerical].corr()
sns.heatmap(data=x,annot=True,linewidths=1, linecolor='white',robust=False)
```

Out[71]:

<Axes: >



Insights and Recommendations:

- Bounce Rates and Administrative pages have high negative correlation which shows that user who spends more time has less chance of leaving the site
- highest correlation between Bounce rates and Exit rates suggest that more the customer leaves the site without visiting other pages is highly likely to exit
- Users who spend more time on specific types of content tend to visit more pages of that content type

Check the distribution of the target variable ('Revenue') to understand class balance.

In [72]:

```
shopping['Revenue'].value_counts(normalize=True)
```

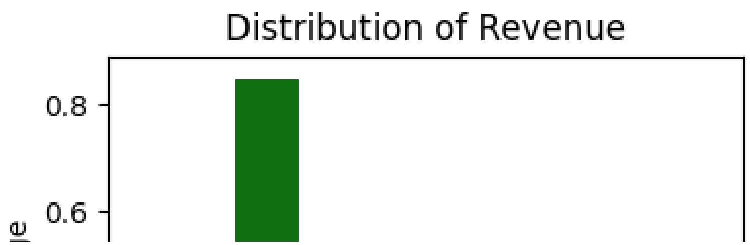
Out[72]:

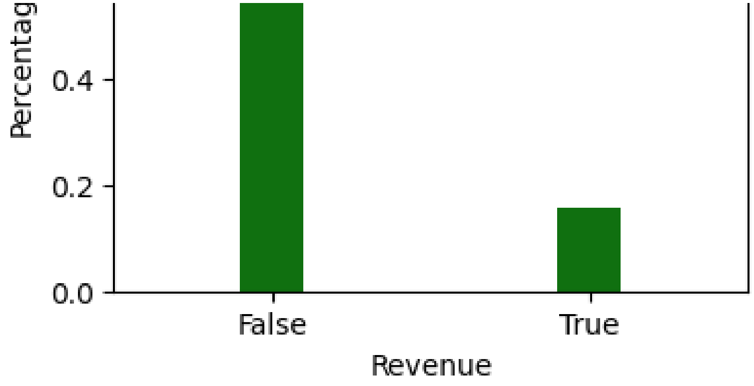
Revenue	proportion
False	0.845255
True	0.154745

dtype: float64

In [73]:

```
Revenue = shopping['Revenue'].value_counts(normalize=True)
plt.figure(figsize=(4,3))
sns.barplot(data=Revenue,width = 0.2,color='green')
plt.title('Distribution of Revenue')
plt.xlabel('Revenue')
plt.ylabel('Percentage')
plt.xticks(rotation=0)
plt.show()
```





Summarize page views, durations, and bounce/exit rates for each page category.

In [74]:

```
page_views = shopping[['Administrative','Informational','ProductRelated']].sum()
avg_views = shopping[['Administrative','Informational','ProductRelated']].mean()
Total_duration = shopping[['Administrative_Duration','Informational_Duration','ProductRelated_Duration']].sum()
Avg_duration = shopping[['Administrative_Duration','Informational_Duration','ProductRelated_Duration']].mean()
Bounce_Rate = shopping['BounceRates'].mean()
Exit_Rate = shopping['ExitRates'].mean()
summary_df = pd.DataFrame({
    'page_views': page_views,
    'avg_views': avg_views,
    'Total_duration': Total_duration,
    'Avg_duration': Avg_duration
})
summary_df
```

Out[74]:

	page_views	avg_views	Total_duration	Avg_duration
Administrative	28546.0	2.315166	NaN	NaN
Administrative_Duration	NaN	NaN	9.964935e+05	80.818611
Informational	6209.0	0.503569	NaN	NaN
Informational_Duration	NaN	NaN	4.250447e+05	34.472398
ProductRelated	391249.0	31.731468	NaN	NaN
ProductRelated_Duration	NaN	NaN	1.473122e+07	1194.746220

Insights and Recommendations:

- Data suggests that high no of page views are for product related pages
- Average duration of product related pages are also high

Analyze SpecialDay distribution and its correlation with Revenue.

In [75]:

```
shopping['SpecialDay'].value_counts()
```

Out[75]:

SpecialDay	count
0.0	11079
0.6	351
0.8	325
0.4	243
0.2	178
1.0	154

dtype: int64

In [76]:

```
shopping['SpecialDay'].corr(shopping['Revenue'])
```

Out[76]:

```
-0.08230459817953266
```

Insights and Recommendations:

- There is no linear relationship between special day and the revenue
- It indicates that special day might not be a major factor

- It indicates that special day might not be a major factor
- Only on further analyses can we truly find out the impact of no of days closer to the special days on revenue

Generate a binary feature indicating whether the user visited all three page categories.

In [77]:

```
s1 = shopping.copy()
s1.head()
s1['Visited_All_Categories'] = s1[['Administrative','Informational','ProductRelated']].apply(lambda x : 'Yes' if all(x > 0) else 'No',axis =1)
s1['Visited_All_Categories'].value_counts()
```

Out[77]:

	count
Visited_All_Categories	
No	10163
Yes	2167

dtype: int64

In [78]:

```
s1.groupby('Visited_All_Categories')['Revenue'].mean().reset_index()
```

Out[78]:

	Visited_All_Categories	Revenue
0	No	0.135983
1	Yes	0.242732

We can observe that the user who has visited all three pages has higher rate of revenue generation

Explore PageValues distribution and its relationship with TrafficType, VisitorType, and Region.

In [79]:

```
shopping['VisitorType'].value_counts()
```

Out[79]:

	count
VisitorType	
Returning_Visitor	10551
New_Visitor	1694
Other	85

dtype: int64

In [80]:

```
pd.crosstab(shopping['TrafficType'],shopping['Region'])
```

Out[80]:

Region	1	2	3	4	5	6	7	8	9
TrafficType									
1	875	240	489	250	80	192	168	101	56
2	1543	330	778	374	121	217	257	140	153
3	881	175	397	200	23	119	108	56	93
4	392	111	209	105	36	75	68	43	30
5	123	21	42	21	6	23	7	3	14
6	148	46	90	39	16	32	38	19	16
7	18	8	7	0	2	1	2	1	1
8	151	25	66	28	5	14	19	14	21
9	22	3	7	3	0	1	5	0	1
10	188	45	86	40	7	22	25	17	20
11	79	32	54	26	3	24	15	8	6

Region	1	2	3	4	5	6	7	8	9
TrafficType									
12	0	0	1	0	0	0	0	0	0
13	272	83	138	75	15	70	35	22	28
14	2	2	2	3	0	2	1	1	0
15	16	0	11	5	0	2	1	3	0
16	2	0	0	0	0	0	0	1	0
17	1	0	0	0	0	0	0	0	0
18	3	1	3	0	0	1	1	1	0
19	10	3	2	0	0	1	1	0	0
20	54	11	21	13	4	9	10	4	72

Investigate user session lengths and their impact on conversion rates.

In [81]:

```
s2 = shopping.copy()
session = ['Administrative_Duration','Informational_Duration','ProductRelated_Duration']
s2['Session_length'] = s2['Administrative_Duration']+s2['Informational_Duration']+s2['ProductRelated_Duration']
s2.groupby('Revenue')['Session_length'].describe()
```

Out[81]:

	count	mean	std	min	25%	50%	75%	max
Revenue								
False	10422.0	1173.964158	1925.426959	0.0	183.666667	588.166667	1464.005253	69921.647230
True	1908.0	2053.304285	2436.111519	0.0	620.358974	1252.075000	2513.856331	28450.201097

In [82]:

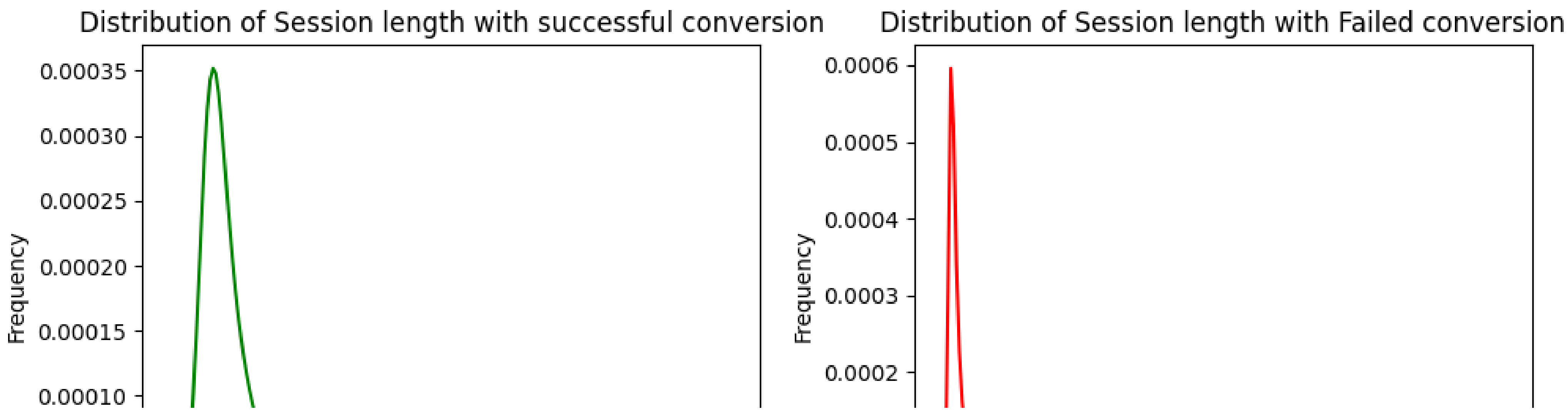
```
Success = s2[s2['Revenue'] == True]['Session_length']
Failure = s2[s2['Revenue'] == False]['Session_length']
n1 = Success.sample(1000)
n2 = Failure.sample(1000)
alpha = 0.05
Ho = 'There is no significant relation between session length and Revenue'
Ha = 'There is a significant relation between session length and Revenue'
```

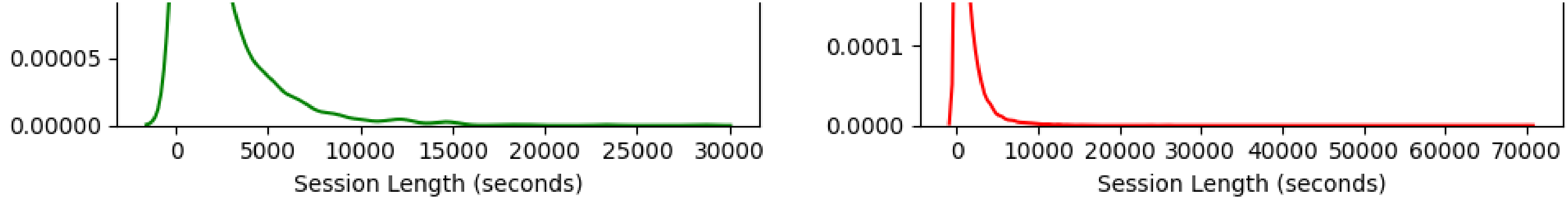
```
statistic, pvalue = ttest_ind(n1,n2)
if pvalue < alpha :
  print(pvalue)
  print('Reject Null Hypothesis')
  print(Ha)
else:
  print(pvalue)
  print('Fail to Reject Null Hypothesis')
  print(Ho)
```

```
5.876197200929111e-26
Reject Null Hypothesis
There is a significant relation between session length and Revenue
```

In [83]:

```
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
sns.kdeplot(Success,color= 'green')
plt.title('Distribution of Session length with successful conversion')
plt.xlabel('Session Length (seconds)')
plt.ylabel('Frequency')
plt.subplot(1,2,2)
sns.kdeplot(Failure,color='red')
plt.title('Distribution of Session length with Failed conversion')
plt.xlabel('Session Length (seconds)')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```





Insights and Recommendations:

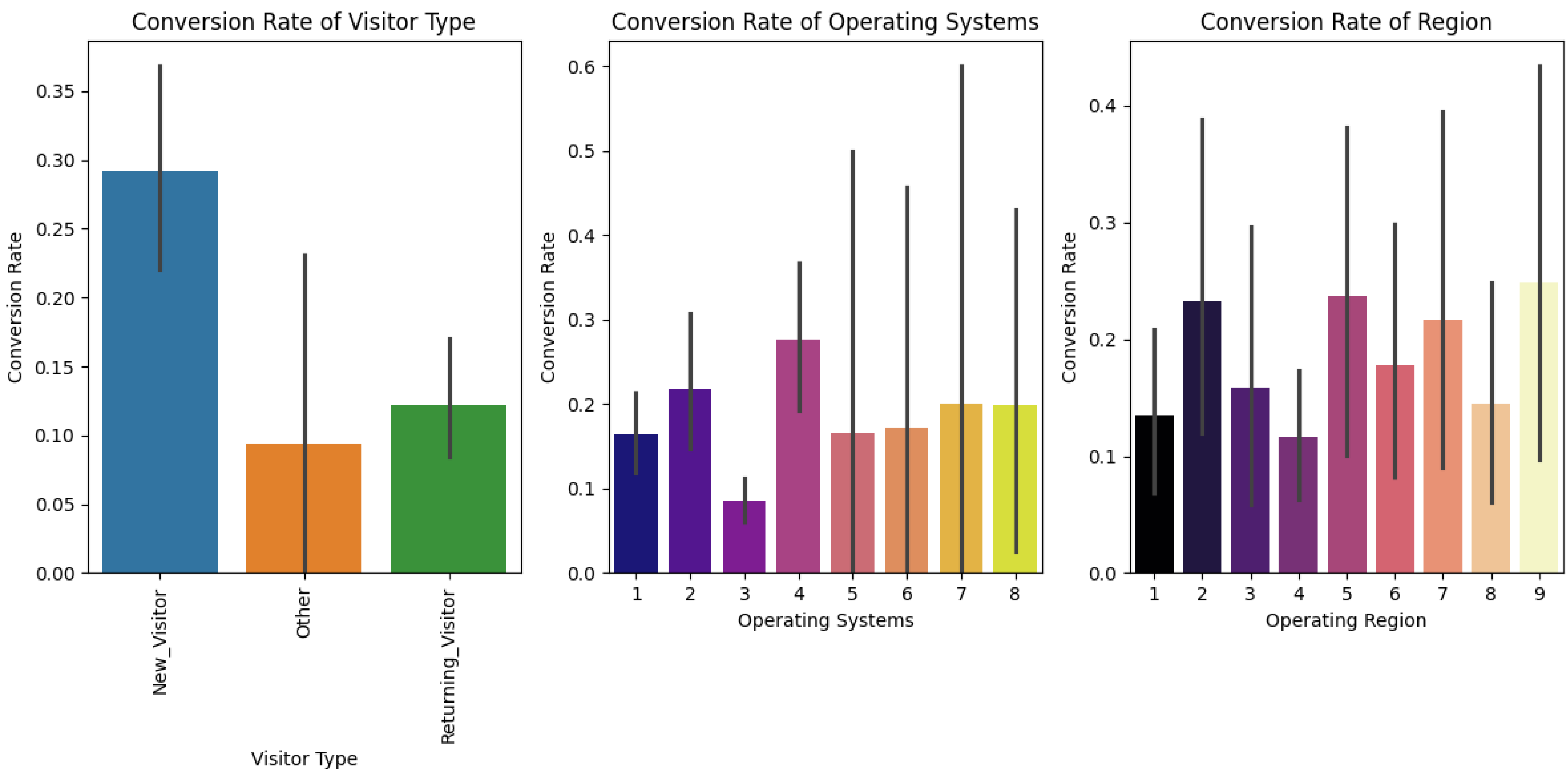
- **Test Data suggests a strong relationship between Session length and conversion rate**
- **Longer sessions are likely associated with higher user engagement, which may involve more interactions with the site, viewing more products, or spending more time considering purchases, all of which can lead to higher revenue.**
- **Encouraging users to spend more time on the site could be an effective strategy to increase sales.**
- **To improve user engagement enhancing the website for user engagement is necessary**

Group users based on VisitorType, OperatingSystems, and Region to identify potential differences in behavior and conversion rates

In [84]:

```
df_grp = s2.groupby(['VisitorType','OperatingSystems','Region']).agg(ConversionRate=('Revenue','mean')).reset_index()

plt.figure(figsize=(12,6))
plt.subplot(1,3,1)
sns.barplot(data=df_grp,x='VisitorType', y='ConversionRate',hue='VisitorType')
plt.title('Conversion Rate of Visitor Type')
plt.xlabel('Visitor Type')
plt.ylabel('Conversion Rate')
plt.xticks(rotation=90)
plt.subplot(1,3,2)
sns.barplot(data=df_grp,x='OperatingSystems', y='ConversionRate',hue='OperatingSystems',palette='plasma')
plt.title('Conversion Rate of Operating Systems')
plt.xlabel('Operating Systems')
plt.ylabel('Conversion Rate')
plt.legend().remove()
plt.subplot(1,3,3)
sns.barplot(data=df_grp,x='Region', y='ConversionRate',hue='Region',palette='magma')
plt.title('Conversion Rate of Region')
plt.xlabel('Operating Region')
```

```
plt.ylabel('Conversion Rate')
plt.legend().remove()
plt.tight_layout()
plt.show()
```



Segment users based on TrafficType and analyze their engagement patterns and purchase probability

In [85]:

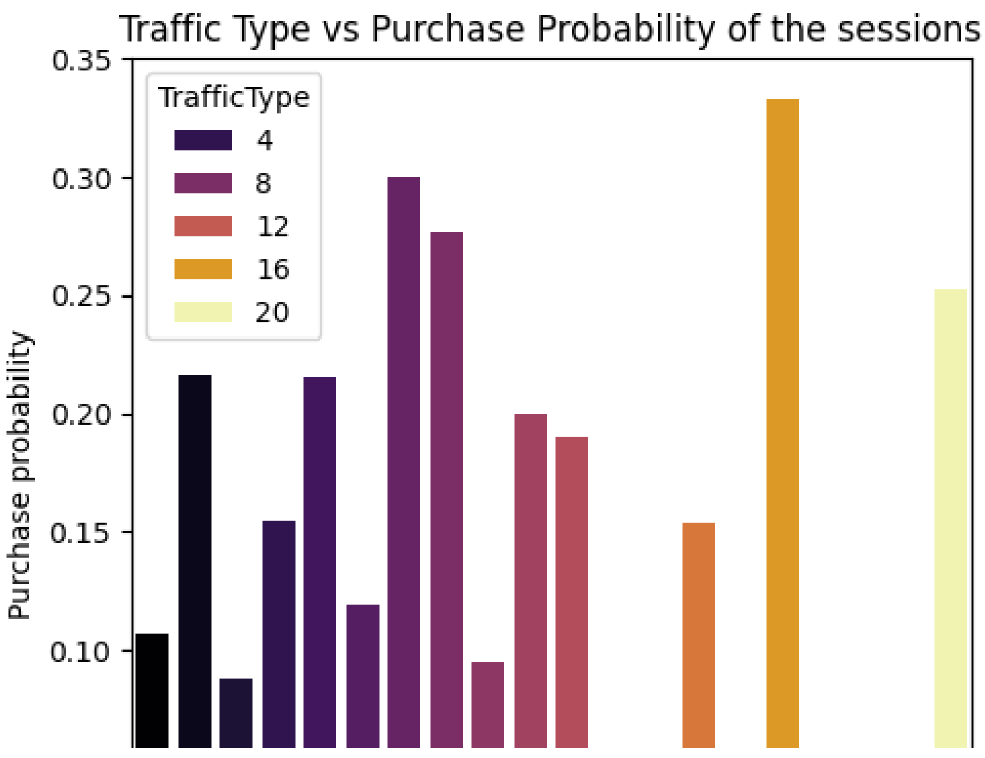
```
df_p = s2.groupby('TrafficType')
[['ProductRelated','ProductRelated_Duration','BounceRates','ExitRates','PageValues','Revenue']].mean().reset_index()
df_p.rename(columns={'Revenue': 'Purchase_Probability'}, inplace=True)
df_p.head()
```

Out[85]:

	TrafficType	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	Purchase_Probability
0	1	31.918401	1234.034177	0.032346	0.055708	3.455074	0.106895
1	2	38.125224	1457.941637	0.008455	0.026391	8.304366	0.216458
2	3	25.805556	892.757712	0.033314	0.057117	3.276033	0.087719
3	4	28.525725	988.944497	0.016261	0.036155	7.043113	0.154350
4	5	17.884615	742.331026	0.009451	0.029679	7.712489	0.215385

In [86]:

```
plt.figure(figsize=(5,5))
sns.barplot(data=df_p,x='TrafficType',y='Purchase_Probability',hue='TrafficType',palette='inferno')
plt.title('Traffic Type vs Purchase Probability of the sessions')
plt.xlabel('Traffic Type')
plt.ylabel('Purchase probability')
plt.tight_layout()
plt.show()
```



In [ ]: